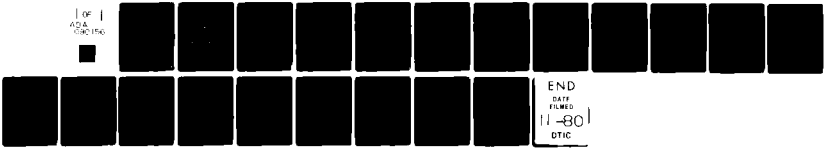
AD-A090 156 COLLEGE OF WILLIAM AND MARY WILLIAMSBURG VA DEPT OF --ETC F/G 9/2
A SURVEY OF BANDWIDTH AND PROFILE REDUCTION ALGORITHMS.(U)
AUG 80 N E GIBBS N00014-76-C-0673
UNCLASSIFIED TR-22 NL



END
DATE FILMED
11-80
DTIC

LEVEL II

(12) B.S.

A Survey of Bandwidth and Profile Reduction Algorithms,

(10) Norman E. Gibbs

(15) N00014-76-C-0673

(14) TR-22

(7) Technical Report 22

The College of William and Mary
Department of Mathematics and Computer Science
Williamsburg, Virginia 23185

(11) Aug [redacted] 80

(12) 20

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Exact Algorithms

In this section it will be assumed that the letter n is a positive integer. If $k \in \{0,1,2,\ldots,n\}$, let $B_{k,n}$ denote the graph on n vertices whose vertices are numbered $(1,2,\ldots,n)$ and whose edge set is $\{(a,b):|a-b|\leq k\}$. Given a graph G on n vertices we can determine its bandwidth as follows (See [10].):

Step 1. Set m <= 0

Step 2. If G is a subgraph of $B_{m,n}$ then stop, m is the bandwidth

Step 3. Set m <= m + 1 and return to Step 2

Unfortunately, this algorithm has little practical significance since determining whether or not a graph is a subgraph of another graph takes a prohibitive amount of time. In 1976, Papadimitriou [18] was able to prove that the bandwidth minimization problem is (in general) NP-complete. In 1978, Garey, Graham, Johnson, and Knuth [7] were able to prove NP-completeness in a sharper form. In the same paper they give a linear algorithm for determining whether or not a graph has bandwidth 2. Gibbs and Poole [10] and Fulkerson and Gross [6] had previously presented linear time algorithms for bandwidth 1. The only gap now is the question of whether or not the problem of:

Is bandwidth(G) $\leq$ 3 NP-complete for arbitrary G?

Production Algorithms

Large sparse matrix problems arise in a variety of application areas, such as structural engineering, fluid dynamics, and network analysis. Although out of vogue for a while, band solvers which solve sparse linear systems of equations directly using Gaussian elimination have been used increasingly on vector processing machines such as the Cray and the CDC Star. (See [15].) Pure Gaussian elimination requires $o(n^3)$ operations for an n x n matrix. The time required to solve the system using a band solver is $o(nb^2)$ where b is the bandwidth. (If A is an n x n matrix its bandwidth is $\max_{a_{ij}\neq 0} |i-j|$.) The problem then becomes how to reduce b as much as possible. Apparently people became adept at labeling by hand the grids (graphs) which the matrices represented. The motivation then existed for the creation of automatic production programs which (hopefully) reduced bandwidth.

In a 1979 survey paper by Everstine [5], 49 "reduction" algorithms are referenced. This list is by no means exhaustive and it is safe to assume that between 50 and 100 reduction algorithms have been designed to date. Although most applications programmers working in the area were not famil-

iar with the results on NP-completeness, hueristic reduction algorithms were developed. Most were designed with the goal of obtaining acceptable (rather than exact) bandwidth.

Alway and Martin published the first bandwidth reduction algorithm in 1965 [1]. The main idea of the algorithm was to examine many row and corresponding column permutations with some rather complex criteria used to discard certain permutations. The authors admit in the paper that the algorithm takes too much time even for graphs of moderate size. Nevertheless, they created an algorithm which was effective for their use in reducing bandwidth of small matrices and stimulated further research.

The first widely used production algorithm was published by Rosen [19] in 1968. The idea behind the algorithm is to first compute the bandwidth of the matrix (graph). The endpoints of the edges which cause the bandwidth to be attained are then examined to see if their vertex labels can be interchanged with other vertex labels. When such row and corresponding column permutations are performed, the bandwidth is recomputed to see if an improvement was made. The entire process iterates until no improvement to bandwidth is made. The algorithm uses a local strategy in terms of the entire structure of the associated graph, and is time consuming. This algorithm was published along with its implementation as a FORTRAN program. Perhaps the most interesting aspect of

the algorithm is that people were still using it in production mode in the late 1970's.

A major breakthrough occurred in 1969 when the famous Cuthill-McKee algorithm [3] was published and subsequently incorporated into NASTRAN (See [4].). To understand the Cuthill-McKee algorithm and the algorithm which follows, it is necessary to define a few terms. The definitions and the next two algorithm descriptions are taken from [11]. A level structure, L(G), of a graph G is a partition of the vertices of the graph into levels $L_1, L_2, \ldots, L_k$ such that

1. all vertices adjacent to vertices in level L_1, are in either level L_1 or L_2,
2. all vertices adjacent to vertices in level L_k are in either level L_k or L_{k-1}

and

3. for $1<i<k$, all vertices adjacent to vertices in level L_i are in either level L_{i-1}, L_i, or L_{i+1}.

To each vertex v of the graph there corresponds a particular level structure $L_v(G)$ called the level structure rooted at v. Its levels are determined by

1. $L_1 = (v)$, and
2. for $i>1$, L_i is the set of all those vertices which are not yet assigned to a level, but are adjacent to ver-

tices of level L_{i-1}.

In any level structure L(G) , rooted or not, w_i (L) = $|L_i|$ is called the width of level i, and $w(L) = \max_i(w_i)$ is the width of the level structure L(G). It is easily observed that for any level structure, L, a numbering of G which assigns consecutive integers level by level, first to the vertices of level L_1 , then to those of level L_2 , and so forth, gives a bandwidth which is less than or equal to 2w(L)-1. Furthermore, if the level structure is rooted, then the bandwidth of the numbering is greater than or equal to w(L). The depth of a level structure is the number of levels, k.

For discussion purposes assume all graphs are connected. The Cuthill-McKee algorithm begins by generating a level structure rooted at each vertex of low degree. See [3] for their definition of low degree. For each rooted level structure of minimal width generated in the first step the graph is numbered level by level with consecutive positive integers by first assigning the integer 1 to the root. For each successive level, starting with level 2, the vertices adjacent to the lowest numbered vertex in the previous level are numbered according to increasing degree with ties broken arbitrarily. The remaining vertices adjacent to the next lower numbered vertex of the preceding level are numbered next, again in order of increasing degree. The process continues

until all vertices of the current level are numbered. This numbering process is repeatedly applied in order to subsequent levels until all vertices have been numbered. The numbering which gives the best bandwidth of any of the rooted level structures of minimal width is selected. This algorithm was the most widely used bandwidth reduction algorithm during the 1970's.

In 1976 Gibbs, Poole, and Stockmeyer published a series of related papers (See [11], [2], and [12].) describing a new bandwidth reduction algorithm which overcame what they perceived to be major shortcomings of the Cuthill-McKee algorithm. The algorithm has three phases, finding a pseudo-diameter, minimizing level width, and numbering a level structure (not necessarily rooted).

The Cuthill-McKee algorithm is inefficient because of the time consumed performing an exhaustive search to find rooted level structures of minimal width. A second problem is that the graph is renumbered and the corresponding bandwidth recomputed, for every level structure found of minimal width. A third problem is that the bandwidth obtained by a Cuthill-McKee numbering can never be less than the width of the rooted level structure used, although the bandwidth of a graph can be less than the width of any rooted level structure. The first two shortcomings are overcome by carefully selecting a starting vertex after generating only a rela-

tively small number of level structures. The graph is renumbered, and corresponding bandwidth computed only once. The third problem is resolved by utilizing a more general type of level structure.

Gibbs, Poole, and Stockmeyer observed that level structures of small width are usually among those of maximal depth. Clearly, increasing the number of levels always decreases the average number of vertices in each level, and tends to reduce the width of the level structure as well. Ideally, then one would like to generate level structures rooted at endpoints of a diameter. Since there is no known efficient procedure which always finds such vertices, the following algorithm is used to find the endpoints of a pseudo-diameter, that is, a pair of vertices that are at nearly maximal distance apart. For a large class of graphs, including all trees, the pseudo-diameter is actually a real diameter.

Finding the endpoints of a pseudo-diameter

A. Pick the smallest numbered vertex of minimal degree and call it v.

B. Generate a level structure L_v rooted at vertex v. Let S be the set of vertices which are in the last level of L (i.e., those vertices which are furthest away from v).

C. Generate level structures rooted at vertices $s \in S$ selected in order of increasing degree. If for some $s \in S$ the

depth of L_s is greater than the depth of L_v , then set v<=s and return to step B.

D. Let u be the vertex of S whose associated level structure has smallest width. The algorithm terminates with u and v the endpoints of a pseudo-diameter.

In the process of finding a pseudo-diameter, this algorithm constructs level structures L_u and L_v rooted at the endpoints u and v, respectively. It is possible to combine these two level structures into a new level structure whose width is usually less than that of either of the original rooted ones, using the following algorithm.

Minimizing Level Width

A. Using the rooted level structures $L_v = (L_1, L_2, \ldots, L_k)$ and $L_u = (M_1, M_2, \ldots, M_k)$ obtained from the algorithm for finding a pseudo-diameter, associate with each vertex w of G the ordered pair (i,j), called the associated level pair, where i is the index of the level in L_v that contains w, and k+1-j is the index of the level in L_u that contains w. Thus the pair (i,j) is associated with a vertex w if and only if $w \in L_i \cap M_{k+1-j}$. Note that the pair (1,1) is associated with the vertex v, while the pair (k,k) is associated with u.

B. Assign the vertices of G to levels in a new level structure $L = (N_1, N_2, \ldots, N_k)$ as follows:

1. If the associated level pair of a vertex w is of the form (i,i), then vertex w is placed in N_i . The vertex

w and all edges incident to w are removed from the graph. If V(G) is empty, stop.

2. The graph G now consists of a set of one or more disjoint connected components $C_1, C_2, \ldots, C_t$ ordered so that $|V(C_1)| \geq |V(C_2)| \geq \ldots \geq |V(C_t)|$.

3. For each connected component (taken in the order $C_1, C_2, \ldots, C_t$) do the following:

a. Compute the vector $(n_1, n_2, \ldots, n_k)$ where $n_i = |N_i|$.

b. Compute the vectors $(h_1, h_2, \ldots, h_k)$ and $(l_1, l_2, \ldots, l_k)$ where $h_i = n_i$ + (the number of vertices which would be placed in N_i if the first element of the associated level pairs were used) and $l_i = n_i$ + (the number of vertices which would be placed in N_i if the second element of the associated level pairs were used).

c. Find $h_0 = \max_i \{h_i : h_i - n_i > 0\}$ and $l_0 = \max_i \{l_i : l_i - n_i > 0\}$.

i. If $h_0 < l_0$, place all the vertices of the connected component in the levels indicated by the first elements of the associated level pairs.

ii. If $l_0 < h_0$, use the second elements of the level pairs to place the vertices in the levels.

iii. If $h_0 = l_0$ then use the elements of the level pairs which arise from the rooted level

structure of smaller width. If the widths are equal, use the first elements.

The algorithm terminates when each vetex of G has been assigned a level in the level structure L.

The numbering procedure is similar to that of the Cuthill-McKee algorithm in that is assigns consecutive positive integers to the vertices of G level by level. A few modifications were necessary, however, since the level structures obtained by the algorithm to minimize level width are of a more general type than the rooted ones used by the Cuthill-McKee algorithm. Under certain conditions, profile can be further reduced by using the reverse numbering described in step D below. (See the next section on profile reduction.)

Numbering

A. If the degree of u is less than the degree of v, then interchange u and v and reverse the level structure obtained by the algorihm to minimize level width by setting N_i to N_{k-i+1}. (This insures that the numbering starts from the end-point of lower degree).

B. Assign consecutive positive integers to the vertices of level N_1 in the following order:

1. Assign the number 1 to the vertex v.
2. Let w be the lowest numbered vertex of level N_1 which has unnumbered vertices in N_1 adjacent to it. Number the vertices of N_1 adjacent to w, in order of

increasing degree. Repeat this step until all vertices of N_1 adjacent to numbered vertices are themselves numbered.

3. If any unnumbered vertices remain in level N_1, number the one of minimal degree, then go to step B.2. Otherwise proceed to step C.

C. Number the vertices of level N_i, i=2,3,...,k as follows:

1. Let w be the lowest numbered vertex of level N_{i-1} that has unnumbered vertices of level N_i adjacent to it. Number the vertices of N_i adjacent to w in order of increasing degree. Repeat this step until all vertices of level N_i adjacent to vertices of level N_{i-1} are numbered.

2. Repeat steps B.2 and B.3, replacing 1 with i.

D. The numbering is reversed by setting i to n-i+1, for i=1,2,...,n if either of the two following conditions holds:

1. Step A interchanged vertices u and v and the algorithm for minimizing level width selected the second elements of the level pairs for component C_1.

2. Step A did not interchange vertices u and v and the algorithm for minimizing level width selected the first elements of the level pairs for component C_1.

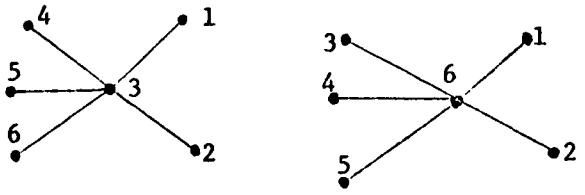
Recently Lewis [15] has made some improvements to the FORTRAN implementation of the Gibbs, Poole, and Stockmeyer algorithm. He claims that this new version will run slightly faster than Algorithm 500 [2].

Profile Reduction

A related but independent problem is that of profile reduction. Let A be an n x n sparse symmetric matrix. To define the profile of A, first define $f_i = \min_i(j : a_{ij} \neq 0)$ for i=1,2,...,n (assume $a_{ii} \neq 0$). This locates the leftmost nonzero element in each row. Next define $p_i = i - f_i$. The profile is defined to be

$$\sum_{i=1}^{n} p_i \quad .$$

Note that in the figure below the first labeling gives a bandwidth of 3 (which is minimal) and a profile of 8. The second labeling gives a bandwidth of 5 and a profile of 5 (which is minimal).



The first practical and widely used profile reduction algorithm was the reverse Cuthill-McKee algorithm which was a modification of the original algorithm due to George [8]. The algorithm is basically the same as described above except a step is added at the end which "reverses" the numbering by replacing i with n-i+1. The reversal of the numbering has no effect on bandwidth and Liu and Sherman [17] have proved that

it can never increase profile.

Other good profile reduction algorithms have been developed by King [14], Gibbs [13], and Snay [20]. Snay's paper contains an exellent description of the intuitive notions of why the King and Cuthill-McKee numberings are good for profile and bandwidth reduction respectively. Of the profile reduction algorithms mentioned, Snay's tends to produce the best (least) profile when it works well. As will be mentioned below, Snay's algorithm sometimes has trouble as does King's on certain graphs. Recently, Lewis [15] modified the FORTRAN implementation of Gibbs' profile reduction algorithm so that it runs almost as fast as the Gibbs, Poole, and Stockmeyer bandwidth reduction algorithm. Although Gibbs' profile reduction algorithm does not always reduce profile as much as Snay's algorithm,Lewis' implementation is a significant contribution in that it has made the more stable algorithm fast enough to be a practical alternative to Snay's algorithm and King's algorithm. Based on percentage of reduction, one can claim that the profile reduction of the three algorithms are comparable.

Evaluation of Reduction Algorithms

Since bandwidth and profile reduction algorithms are hueristic, it is difficult to compare them. An approach

taken by Gibbs, Poole, and Stockmeyer [12] has gained in popularity and has probably defined the de facto standard for reduction algorithm testing. Dr. Gordon Everstine of the David Taylor Naval Ship and Research Development Center in Bethesda, Maryland (DTNSRDC) provided them with 19 test matrices which arise when the finite element method is used to approximate solutions of partial differential equations in several structural engineering applications. The structures included parts of aircraft, gasoline storage tanks, submarines, propellor blades, and satellites. More recently, Everstine [5] assembled a new set of application matrices. This set of 30 matrices includes some of the original 19 and drawings of the structutes represented by the matrices are available. This set of matrices is now the standard benchmark for any production reduction algorithm.

In the same paper, Gibbs, Poole, and Stockmeyer ran a second set of tests. They were interested in obtaining timing data in order to estimate the asymptotic behavior of the algorithms they were testing. The algorithms were tested on several families of grids. The grids selected were n x n squares, 3n x n rectangles, n x 20n rectangles, 20 n x n cylinders, and n x 20n cylinders. (See the paper for more details about the types of elements used in the families of grids.) George and Liu adopted the same technique in Liu's thesis [16] and in a later paper [9].

The problem of how to evaluate a heuristic algorithm has no easy solution. It is usually not difficult to come up with bizarre counterexamples which will destroy the performance of almost any reduction algorithm -- good or bad. Experience has shown, however, that the results of testing algorithms on the set of 19 test matrices and then on the set of 30 test matrices correlate well. These results have led a number of researchers to believe that the set of 30 test matrices has enough variety to be representative of typical application problems which would require the use of a reduction algorithm. An interesting example throughout testing reduction algorithms has been the matrix with 918 vertices. The Snay algorithm, the King algorithm, and the reverse Cuthill-McKee algorithm have all performed poorly on it independent of starting points. In fact there is no starting point for which Snay's algorithm or King's algorithm performs well on this example. The reason why is an open problem.

REFERENCES

[1] Alway, G. G. and Martin, D. W., "An algorithm for reducing the bandwidth of a matrix of symmetrical configuration", Computer Journal, 8 (1965), 264-272.

[2] Crane, H. L., Jr., Gibbs, N. E., Poole, W. G., Jr., and Stockmeyer, P. K., "Algorithm 508 -- Matrix bandwidth and profile reduction", ACM Transactions on Mathematical Software, 2 (1976), 375-377.

[3] Cuthill, E. and McKee, J. M., "Reducing the bandwidth of sparse symmetric matrices", Proceedings of the 24'th National Conference, Association for Computing Machinery, ACM Publication P69, New York, (1969), 157-172.

[4] Everstine, G., "The BANDIT computer program for the reduction of matrix bandwidth for NASTRAN", Naval Ship Research & Development Center, Bethesda, Maryland, Report 3827, (1972).

[5] Everstine, G. C., "A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront", International Journal for Numerical Methods in Engineering, 14 (1979), 837-863.

[6] Fulkerson, D. R. and Gross, O. A., "Incidence matrices and interval graphs", Pacific Journal of Mathematics, 15 (1965), 835-855.

[7] Garey, M. R., Graham, R. L., Johnson, D. S. and Knuth, D. E., "Complexity results for bandwidth minimization", SIAM Journal for Applied Mathematics, 34 (1978), 477-495.

[8] George, J. A., "Computer implementation of the finite element method", Ph. D. Dissertation, Technical Report STAN-CS-71-208, Computer Science Department, Stanford University, Stanford, California, (1971).

[9] George, J. A. and Liu, J. W. H., "Algorithms for matrix partitioning and the numerical solution of finite element systems", SIAM Journal of Numerical Analysis, 15 (1978), 297-327.

[10] Gibbs, N. E. and Poole, W. G., Jr., "Tridiagonalization by permutations", Communications of the ACM, 20 (1974), 20-24.

[11] Gibbs, N. E., Poole, W. G., Jr. and Stockmeyer, P. K., "An algorithm for reducing the bandwidth and profile of a

sparse matrix", SIAM Journal on Numerical Analysis, 13 (1976), 235-251.

[12] Gibbs, N. E., Poole, W. G., Jr. and Stockmeyer, P. K., "A comparison of several bandwidth and profile reduction algorithms", ACM Transactions on Mathematical Software, 2 (1976), 322-330.

[13] Gibbs, N. E., "Algorithm 509 - A hybrid profile reduction algorithm", ACM Transactions on Mathematical Software, 2 (1976), 378-387.

[14] King, I. P., "An automatic reordering scheme for simultaneous equations derived from network systems", International Journal for Numerical Methods in Engineering, 2 (1970), 523-533.

[15] Lewis, J. G., "Implementation of the Gibbs-Poole-Stockmeyer algorithm and the Gibbs-King algorithm (Algorithms 508 and 509)", Private Communication, (1980).

[16] Liu, J. W. H., "On reducing the profile of sparse symmetric matrices", Ph. D. Dissertation, University of Waterloo, Ontario, Canada, Department of Computer Science -- Faculty of Mathematics, (1975).

[17] Liu, J. W. H. and Sherman, A. H., "Comparative analysis of the Cuthill-McKee ordering algorithms for sparse matrices", SIAM Journal on Numerical Analysis, 13 (1976), 197-213.

[18] Papadimitriou, Ch. H., "The NP-completeness of the bandwidth minimization problem", Computing, 16 (1976), 263-270.

[19] Rosen, R., "Matrix bandwidth minimization", Proceedings of the 23rd National Conference of the ACM, Brandon Systems Press, Princeton, N. J., (1968), 585-595.

[20] Snay, R. A., "Reducing the profile of sparse symmetric matrices", Bulletin Geodesique, 50 (1976), 341-352.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

AD-A090 156

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) College of William and Mary Department of Mathematics & Computer Science Williamsburg, Virginia 23185	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

A Survey of Bandwidth and Profile Reduction Algorithms

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report 22, 1980 (August)

5. AUTHOR(S) (First name, middle initial, last name)

Norman E. Gibbs

6. REPORT DATE August 1980	7a. TOTAL NO. OF PAGES 20	7b. NO. OF REFS 20
8a. CONTRACT OR GRANT NO. ONR Contract N00014-76-C-0673 b. PROJECT NO. NR 044-459 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Report 22 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. DISTRIBUTION STATEMENT

Approved for public release;
distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Mathematics Program Office of Naval Research Arlington, Virginia 22217

13. ABSTRACT

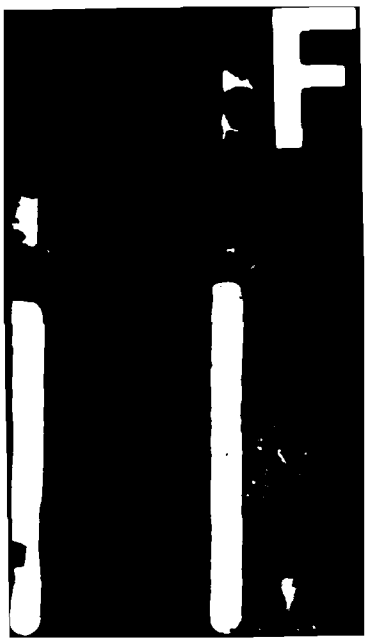
As of 1980 over 50 bandwidth and/or profile reduction algorithms have appeared in the literature. This paper traces the development of the major reduction algorithms and discusses how automatic reduction algorithms can be compared and evaluated.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Bandwidth						
Profile						
Reduction Algorithms						
Finite Element Method						


F